Numerical solution for initial and boundary value problems of high-order ordinary differential equations using Hermite neural network algorithm with improved extreme learning machine

Yanfei Lu
Taizhou University
Futian Weng
Xiamen University
Hongli Sun (honglisun@126.com)
Central South University

Research Article

Keywords: algorithm, High-order ODEs, High-order SODEs, Closed form approximate solution, IELM

Posted Date: October 20th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-818207/v1

Noname manuscript No.
(will be inserted by the editor)

Numerical solution for initial and boundary value problems of high-order ordinary differential equations using Hermite neural network algorithm with improved extreme learning machine

Yanfei Lu, Futian Weng, Hongli Sun*

Received: date / Accepted: date

Abstract In this paper we put forth Hermite neural network (HNN) algorithm with improved extreme learning machine (IELM) to solve initial/boundary value problems of high-order ordinary differential equation(ODEs) and high-order system of ordinary differential equations (SODEs). The model function was expressed as a sum of two terms, where the first term contains no adjustable parameters but satisfies the initial/boundary conditions, the second term involved a single-layered neural network structure with IELM and Hermite basis functions to be trained. The approximate solution is presented in closed form by means of HNN algorithm, whose parameters are obtained by solving a system of linear equations utilizing IELM, which reduces the complexity of the problem. Numerical results demonstrate that the method is effective and reliable for solving high-order ODEs and high-order SODEs with initial and boundary conditions.

Keywords HNN algorithm; High-order ODEs; High-order SODEs; Closed form approximate solution; IELM

Mathematics Subject Classification (2020) 34A30 ; 65D15

1 Introduction

It is well known that a large number of research projects, carried out in physics, chemistry, biology, economics, make use of mathematical models partially or fully

Yanfei Lu
School of Electronics and Information Engineering, Taizhou University, Taizhou, Zhejiang 318000, China,
E-mail: luyanfei1216@sina.com.

Futian Weng
Data Mining Research Center, Xiamen University, Xiamen, Fujian 361005, China,
E-mail: wengfutian@gmail.com.

Hongli Sun
School of Mathematics and Statistics, Central South University, Changsha, Hunan 410083, China,
E-mail: honglisun@126.com.

formulated by ordinary differential equations (ODEs) and system of ordinary differential equations (SODEs) with higher order [1], [2]. However, given the complexity of the proposed models, their makers are often faced with difficult problems and unknown analytic solutions. Many researchers studied numerical methods for solving high-order ODEs and high-order SODEs. Depending upon the form of the boundary conditions to be satisfied by the solution, problems involving in such equations can be divided into two main classes, namely boundary value problems (BVPs) and initial value problems (IVPs).

Numerous works on the IVPs and BVPs of high-order ODEs have been published in the last years. Most of these investigations are focused on two main aspects: one is to simplify the higher order ODE into an equivalent system of first-order ODEs, which involves a lot of computer time and more human effort. The other is to solve directly high-order ODEs, such as Block methods [3], Runge-Kutta methods [4], [5], Classical orthogonal polynomials (Legendre polynomials, Chebyshev polynomials, Jacobi polynomials), Wavelet methods (Haar wavelets [6],[7], Shannon wavelets [8]), Adomian decomposition method with Green's function [9], spectral algorithms [10]-[12], etc. However, most of these numerical methods provide a discrete solution or a solution of limit differentiability.

Concerning IVPs and BVPs of high-order SODEs, various numerical techniques to compute the numerical solutions have been developed. For instance, Xiong selected direct Runge-Kutta-type method for solving IVPs of third-order SODEs [13]. Dehghan applied the local radial basis functions based differential quadrature collocation method to solve BVPs of second-order ordinary differential systems [14]. Caglar considered B-spline method to solve linear system of second-order boundary value problems [15].

Due to the rapid development of computer hardware and various new theories, neural networks and other machine intelligence algorithms are gradually applied to various areas, such as pattern recognition [16], classification problem, forecasting [17], image processing and so on. In the field of solving differential equation, various intelligent algorithms, in particular artificial neural networks (ANNs), are often superior to the traditional methods. The neural network (NN) approaches for solving differential equations mainly include the following categories: multilayer perceptron NN, radial basis function NN, cellular NN, finite element NN and wavelet NN. The neural network methods have been also used in finding the solution of high–order ODEs and high-order SODEs. Mai-Duy presented alternative high order methods based on radial basis function networks for solving ODEs directly [18]. A hybrid neural network based on optimization technique is used to solve IVPs and BVPs of high-order ODEs [19]. Yazdi et al. developed an unsupervised version of KLMS algorithm along with new optimization approach for solving first and second-order ODEs [20]-[22]. Chakraverty and Mall proposed a single-layer Legendre neural network and a single-layer Chebyshev neural network to solve the IVPs and BVPs of higher order ODEs [23],[24]. Yanfei et al. introduced the improved LS-SVM algorithm for solving two-point and multi-point BVPs of high-order linear and nonlinear ODEs [25].

Guangbin et al. [26] proposed the extreme learning machine (ELM) algorithm. ELM algorithm provides faster learning speed, better generalization performance and

least human intervention [28,27,29]. Depending upon these advantages, ELM algorithm has been successfully applied to solve ordinary differential equations, partial differential equations and integro-differential equations [34,30,35,31–33]. Among others, Yunlei et al. [34,35] proposed Legendre Neural Network algorithm to solve ODEs and elliptic PDEs. Hongli et al. [36] introduced Bernstein Neural Network algorithm based on ELM and Bernstein polynomial to solve directly first order ODEs, second order ODEs and partial differential equations(PDEs). Yangjin and Yanfei respectively proposed trigonometric neural network algorithm and Legendre Neural Network algorithm for solving the ruin probabilities in the classical risk model and the Erlang(2) risk model [37], [38]. Yinghao [39] proposed a numerical method of the generalized Black-Scholes differential equation using Laguerre neural network.

In this article, we propose a neural network algorithm based on Hermite improved extreme learning machine to directly solve IVPs and BVPs of high-order ODEs and high-order SODEs. In contract with the traditional methods, the approximate solution obtained from HNN algorithm is is available in the form of a continuous and differentiable function, and the algorithm does not need to reduce high-order differential equations to a system of first-order differential equations. The product of two Hermite polynomials is chosen as basis function of hidden neurons, the approximate solution and its derivatives are expressed by using Hermite network. The model function was expressed as a sum of two terms. The first term, which contains no adjustable parameters, was related to the initial/boundary conditions and the second term involved a single-layered neural network structure with extreme learning machine and Hermite basis functions to be trained. Furthermore, HNN algorithm adopts the Moore-Penrose generalized inverse operation to obtain the output weights of network, which can effectively overcomes the drawbacks of the traditional BP neural network, and have higher simulation accuracy.

The remainder of this paper is organized as follows: Section 2 discusses the Hermite neural network model. We describe the form and construction of the solution of IVPs and BVPs of high-order ODEs and high-order SODEs in Section 3. In Section 4, we present numerical experiments to exhibit the effectiveness and superiority of the HNN algorithm for solving IVPs and BVPs of high-order SODEs. Finally, concluding remarks are presented in Section 5.

2 The Hermite neural network model

Hermite neural network model consists of an input node, a functional expansion block based on Hermite polynomials and an output node. Figure 1 shows the architecture of Hermite neural network algorithm.

The polynomials orthogonal with respect to the weighting function e^{-x^2} are the Hermite polynomials [40]. They can be defined by the formula

$$H_j(x) = (-1)^j e^{x^2} \frac{d^j}{dx^j}\left(e^{-x^2}\right). \tag{1}$$

It is easy to check that $H_j(x)$ is a polynomial of degree j.

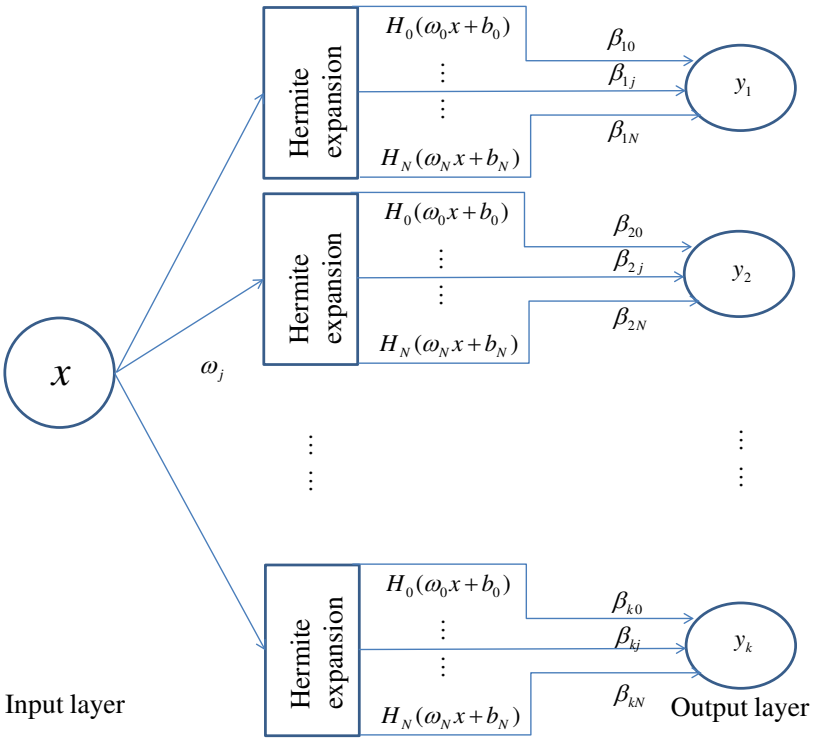

Fig. 1: Architecture of Hermite neural network algorithm

The Hermite polynomials have orthogonality property, given by

$$\int_{-\infty}^{+\infty} e^{-x^2} H_p(x) H_j(x) dx = \begin{cases} 0, p \neq j; \\ 2^j j! \sqrt{\pi}, p = j. \end{cases} \tag{2}$$

These polynomials satisfy a recursive relationship given by

$$\begin{aligned} &H_0(x) = 1, \\ &H_1(x) = 2x, \\ &H_{j+1}(x) = 2xH_j(x) - 2jH_{j-1}(x), j \geq 1. \end{aligned} \tag{3}$$

The Hermite basis functions are differentiable. Their derivatives are also differentiable, and satisfy

$$H_j'(x) = 2xH_j(x) - H_{j+1}(x), j = 0,1,2,...,N. \tag{4}$$

Differentiate this equation again to get

$$H_j''(x) - 2xH_j'(x) + 2jH_j(x) = 0, j = 0,1,2,...,N. \tag{5}$$

Parameters:
(1) L: Number of the collocation points;
(2) N: Number of the hidden neurons;
(3) $\widetilde{\Theta}$: The hidden layer output matrix;
(4) β_{kj}: The output weights.
Steps:
Step 1: The interval $[a,b]$ is discretized into a series of collocation points
$$\Omega = \{a = x_0 < x_1 < ... < x_L = b\}, x_l = a + (b-a)l/L, l = 0,1,...,L.$$
Step 2: Assume the approximate solution by using Hermite polynomials as an activation function, that is, $y_k(x) = Z_k(x) + M(x)\psi_k(x,\omega_j,b_j)$, where $Z_k(x)$ is related to the initial or boundary conditions and $\psi_k(x,\omega_j,b_j)$ involved a single-layered neural network structure.
Step 3: Calculate the partial derivative of $y_k(x)$ and substitute it into the high-order ODEs and high–order SODEs to obtain a system of linear equations.
Step 4: Solving the system of linear equations obtained in step 3 by using ELM algorithm. The samples $x_l, l = 0,1,2,...,L$ will be used as the inputs of the neural networks, and training the neural network is simply equivalent to finding a least squares solution $\widehat{\beta}_{kj}$ of the linear system $\widetilde{\Theta}_k\beta_{kj} = G_k$:
$$\|\widetilde{\Theta}_k\widehat{\beta}_{kj} - G_k\| = \min_{\beta_{kj}} \|\widetilde{\Theta}_k\beta_{kj} - G_k\|,$$
where $\widetilde{\Theta}_k = [\Theta_{lj}]_{L+1,N+1}, k = 1,2,...,M$.
Step 5: The weights β_{kj} from hidden layer to output layer can be obtained. From the theory of Moore–Penrose generalized inverse of matrix $\widetilde{\Theta}_k$, we have
$$\beta_{kj} = \widetilde{\Theta}_k^{\dagger} G_k,$$
where the minimum norm least-square solution is unique and has the smallest norm among all the least-square solutions.
Step 6: We can find the output of the HNN model of high-order ODEs and high-order SODEs.

Table 1: Steps of Hermite neural network model

Furthermore, we can obtain $H'(x) = H(x)Q$, where Q is the Hermite operational matrix given by

$$Q = \begin{bmatrix} 0 & 2 & 0 & 0 & 0 & 0 & ... & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & ... & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 & ... & 0 & 0 \\ 0 & 0 & 0 & 0 & 8 & 0 & ... & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & ... & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & ... & 0 & 2N \\ 0 & 0 & 0 & 0 & 0 & 0 & ... & 0 & 0 \end{bmatrix}_{(N+1)\times(N+1)}, \quad (6)$$

$H(x) = [H_0(x), H_1(x), ..., H_N(x)]$ and $H'(x) = [H'_0(x), H'_1(x), ..., H'_N(x)]$.

Using these basis functions, the approximate solution to high-order ordinary differential equation can be written as

$$\psi_k(x, \omega_j, b_j) = \sum_{j=0}^{N} \beta_{kj} H_j(\omega_j x + b_j), k = 1, 2, ..., M, \tag{7}$$

where x is the input value of the Hermite neural network algorithm, $\psi_k(x, \omega_j, b_j)$ is the output value. ω_j is the weight of the input node to the jth hidden node, b_j is the threshold for the jth hidden node, β_{kj} is the weight vector of the jth hidden node to the output node. The steps of Hermite neural network model for solving high-order ODEs and high-order SODEs are in Table 1.

For simplicity, we take $\omega_j = 1, b_j = 0$ in the following discussion. So the model becomes

$$\psi_k(x, \omega_j, b_j) = \sum_{j=0}^{N} \beta_{kj} H_j(x), k = 1, 2, ..., M. \tag{8}$$

3 Formulation of the method

This section we introduce the form and construction of the solution of IVPs and BVPs of the high-order ODEs and high-order SODEs using HNN algorithm. The model function was expressed as a sum of two terms. The first term, which contains no adjustable parameters, is related to the initial or boundary conditions and the second term involved a single-layered neural network structure with extreme learning machine and Hermite basis functions to be calculated.

3.1 Formulation of the method for IVP in high-order ODEs

The general form of high-order ODEs is

$$\frac{d^M y(x)}{dx^M} = f\Big(x, y, \frac{dy}{dx}, ..., \frac{d^{M-1} y}{dx^{M-1}}\Big), \quad M \geq 2, \; x \in [a, c], \tag{9}$$

subject to initial conditions $y^{(m)}(a) = A_m, m = 0, 1, ..., M-1$.

Corresponding HNN trial solution may be written as [19]

$$y(x) = Z(x) + M(x)\psi(x, \omega_j, b_j) \tag{10}$$

where $Z(x)$ and $M(x)$ are general form of functions used to construct $y(x)$. $M(x)$ is chosen as $M(x) = (x-a)^M$, and $Z(x) = \sum_{n=0}^{M-1} u_n x^n$ is the general polynomial of degree $M-1$, where $\{u_n\}_{n=0}^{M-1}$ are the solution to the upper triangle system of M linear equations in the form

$$\sum_{n=m}^{M-1} \binom{m}{n} m! a^{n-m} u_n = A_m, m = 0, 1, ..., M-1.$$

3.2 Formulation of the method for BVP in high-order ODEs

Let us assume that the high-order ODEs with BVP has the following form

$$\frac{d^M y(x)}{dx^M} = f\Big(x, y, \frac{dy}{dx}, ..., \frac{d^{M-1}y}{dx^{M-1}}\Big), \quad M \geq 2,\ x \in [a,c], \tag{11}$$

subject to boundary conditions $y^{(s)}(a) = p_s, y^{(r)}(c) = q_r,\ 0 \leq s \leq S,\ 0 \leq r \leq R,\ R+S = M-2$.

HNN trial solution for the high-order ODEs with these boundary conditions is formulated as

$$y(x) = Z(x) + M(x)\psi(x, \omega_j, b_j) \tag{12}$$

where $Z(x)$ and $M(x)$ are general form of functions used to construct $y(x)$.

The trial solution satisfies following relations

$$\begin{cases} Z(a) = p_0, \\ Z(c) = q_0, \\ Z'(a) = p_1, \\ Z'(c) = q_1, \\ ..... \\ Z^{(S)}(a) = p_S, \\ Z^{(R)}(c) = q_R, \end{cases} \quad and \quad \begin{cases} M(a)\psi(a, \omega_j, b_j) = 0, \\ M(c)\psi(c, \omega_j, b_j) = 0, \\ M'(a)\psi(a, \omega_j, b_j) + M(a)\psi'(a, \omega_j, b_j) = 0, \\ M'(c)\psi(c, \omega_j, b_j) + M(c)\psi'(c, \omega_j, b_j) = 0, \\ ..... \\ (M(a)\psi(a, \omega_j, b_j))^{(S)} = 0, \\ (M(c)\psi(a, \omega_j, b_j))^{(R)} = 0. \end{cases} \tag{13}$$

The function $M(x)$ is chosen as $M(x) = (x-a)^{M/2}(x-c)^{M/2}$ which satisfies the set of equations in (11) . Here, $Z(x) = t_{M-1}x^{M-1} + t_{M-2}x^{M-2} + ... + t_1 x + t_0$ is the general polynomial of degree $M-1$, where $t_{M-1}, t_{M-2}, ..., t_0$ are constants. From the set of equations, we can have

$$\begin{cases} t_{M-1}a^{M-1} + t_{M-2}a^{M-2} + ... + t_1 a + t_0 = p_0, \\ t_{M-1}c^{M-1} + t_{M-2}c^{M-2} + ... + t_1 c + t_0 = q_0, \\ (M-1)t_{M-1}a^{M-2} + (M-2)t_{M-2}a^{M-3} + ... + t_1 = p_1, \\ (M-1)t_{M-1}c^{M-2} + (M-2)t_{M-2}c^{M-3} + ... + t_1 = q_1, \\ ..... \\ P^S_{M-1}t_{M-1}a^{M-S-1} + P^S_{M-2}t_{M-2}a^{M-S-2} + ... + P^S_S t_S = p_S, \\ P^R_{M-1}t_{M-1}c^{M-S-1} + P^R_{M-2}t_{M-2}c^{M-S-2} + ... + P^R_R t_R = q_R, \end{cases} \tag{14}$$

where P^s_n and P^r_n are number of permutations.

3.3 Formulation of the method for IVP in high-order SODEs

Consider the initial value problems in high-order SODEs of the following form

$$\frac{d^M y_k(x)}{dx^M} = f_k\Big(x, y_1, ..., y_k, \frac{dy_1}{dx}, ..., \frac{dy_k}{dx}, ..., \frac{d^{M-1}y_1}{dx^{M-1}}, ..., \frac{d^{M-1}y_k}{dx^{M-1}}\Big), \quad M \geq 2,\ x \in [a,c], \tag{15}$$

subject to initial conditions $y_k^{(m)}(a) = A_{mk}, k = 1,2,...,M, m = 0,1,...,M-1$.

Corresponding HNN trial solution may be written as [19]

$$y_k(x) = Z_k(x) + M(x)\psi(x, \omega_j, b_j) \tag{16}$$

where $Z_k(x)$ and $M(x)$ are general form of functions used to construct $y_k(x)$. $M(x)$ is chosen as $M(x) = (x-a)^M$, and $Z_k(x) = \sum_{n=0}^{M-1} u_{nk}x^n$, where $\{u_{nk}\}_{k=1}^{M}$ for $(n = 0,1,...,M-1)$ are the solution to the upper triangle system of M linear equations in the form

$$\sum_{n=m}^{M-1} \binom{m}{n} m! a^{n-m} u_{nk} = A_{mk}, k = 1,2,...,M, m = 0,1,...,M-1.$$

3.4 Formulation of the method for BVP in high-order SODEs

Boundary value problem of high-order SODEs to be solved can be stated as follows

$$\frac{d^M y_k(x)}{dx^M} = f_k\Big(x, y_1, ..., y_k, \frac{dy_1}{dx}, ..., \frac{dy_k}{dx}, ..., \frac{d^{M-1}y_1}{dx^{M-1}}, ..., \frac{d^{M-1}y_k}{dx^{M-1}}\Big), \quad M \geq 2,\ x \in [a,c], \tag{17}$$

subject to boundary conditions $y_k^{(s)}(a) = p_{sk}, y_k^{(r)}(c) = q_{rk}, \quad 0 \leq s \leq S, \quad 0 \leq r \leq R, \ R = M-2-S$.

HNN trial solution for the high-order SODEs is formulated as

$$y_k(x) = Z_k(x) + M(x)\psi_k(x, \omega_j, b_j), \tag{18}$$

where $Z_k(x)$ and $M(x)$ are general form of functions used to construct $y_k(x)$.

The trial solution satisfies following relations

$$\begin{cases} Z_k(a) = p_{0k}, \\ Z_k(c) = q_{0k}, \\ Z_k'(a) = p_{1k}, \\ Z_k'(c) = q_{1k}, \\ ..... \\ Z_k^{(S)}(a) = p_{Sk}, \\ Z_k^{(R)}(c) = q_{Rk}, \end{cases} \quad and \quad \begin{cases} M(a)\psi_k(a, \omega_j, b_j) = 0, \\ M(c)\psi_k(c, \omega_j, b_j) = 0, \\ M'(a)\psi(a, \omega_j, b_j) + M(a)\psi_k'(a, \omega_j, b_j) = 0, \\ M'(c)\psi(c, \omega_j, b_j) + M(c)\psi_k'(c, \omega_j, b_j) = 0, \\ ..... \\ (M(a)\psi_k(a, \omega_j, b_j))^{(S)} = 0, \\ (M(c)\psi_k(a, \omega_j, b_j))^{(R)} = 0. \end{cases} \tag{19}$$

The function $M(x)$ is chosen as $M(x) = (x-a)^{M/2}(x-c)^{M/2}$ which satisfies the set of equations in (17). Here, $Z_k(x) = t_{M-1}x^{M-1} + t_{M-2}x^{M-2} + ... + t_1x + t_0$ is the

general polynomial of degree $M-1$, where $t_{M-1}, t_{M-2}, ..., t_0$ are constants. From the set of equations, we can have

$$\begin{cases} t_{M-1}a^{M-1} + t_{M-2}a^{M-2} + ... + t_1 a + t_0 = p_{0k}, \\ t_{M-1}c^{M-1} + t_{M-2}c^{M-2} + ... + t_1 c + t_0 = q_{0k}, \\ (M-1)t_{M-1}a^{M-2} + (M-2)t_{M-2}a^{M-3} + ... + t_1 = p_{1k}, \\ (M-1)t_{M-1}c^{M-2} + (M-2)t_{M-2}c^{M-3} + ... + t_1 = q_{1k}, \\ ..... \\ P^S_{M-1}t_{M-1}a^{M-S-1} + P^S_{M-2}t_{M-2}a^{M-S-2} + ... + P^S_S t_S = p_{Sk}, \\ P^R_{M-1}t_{M-1}c^{M-S-1} + P^R_{M-2}t_{M-2}c^{M-S-2} + ... + P^R_R t_R = q_{Rk}, \end{cases} \tag{20}$$

where P^s_n and P^r_n are number of permutations.

4 Numerical experiments

In this section, to demonstrate the feasibility and superiority of the proposed Hermite neural network algorithm, we carry out seven numerical experiments for IVPs and BVPs of higher order linear ODEs and higher order linear SODEs. We may take the maximum absolute error (MAE) and mean square error (MSE) to measure the accuracy and efficiency of HNN algorithm. They can be defined as follows:

$$\begin{aligned} MAE &= max\{\Delta m_0, \Delta m_1, ..., \Delta m_L\}, \\ MSE &= \frac{1}{L+1}\sum_{l=0}^{L} \Delta m_l^2, \end{aligned} \tag{21}$$

where $\Delta m_l = \mid y(x_l) - \hat{y}(x_l) \mid, l = 0, 1, ..., L$. Moreover, let $y(x_l)$ denote the close solution of the higher order linear ODEs and higher order linear SODEs, and $\hat{y}(x_l)$ represent the approximate solution obtained by the proposed algorithm. In particular, our method is compared with LS-SVM [25], non-polynomial spline [41] and Runge-Kutta (R-K) method and a detailed analysis is made.

4.1 Example 1

Consider a fourth-order linear ODE with boundary conditions [25]

$$\begin{cases} \frac{d^4y}{dx^4} + y(x) = \Big((\frac{\pi}{2})^4 + 1\Big)cos\Big(\frac{\pi}{2}x\Big), \quad x \in [-1, 1], \\ y(-1) = 0, \\ y(1) = 0, \\ y'(-1) = \pi/2, \\ y'(1) = -\pi/2. \end{cases} \tag{22}$$

The analytical solution is $y = cos\big(\pi x/2\big)$.

Twenty-one equidistant points are used for the training points and the first eight order Hermite polynomials are used as basis functions. The approximate solution obtained by the proposed HNN algorithm is compared with the exact solution in Fig. 2 (a) and the errors are reported in Fig. 2 (b). From the obtained results, we can calculate that the MSE is approximately 9.77×10^{-23}. Although training was performed applying just a small number of samples in the domain $[-1,1]$, the errors remain high accuracy, which confirms the reliability of this method. Besides, we have solved this problem by the proposed HNN algorithm with $L = 20, N = 5$, $L = 20, N = 10$ and $L = 100, N = 10$, the errors are reported in Tab. 9.

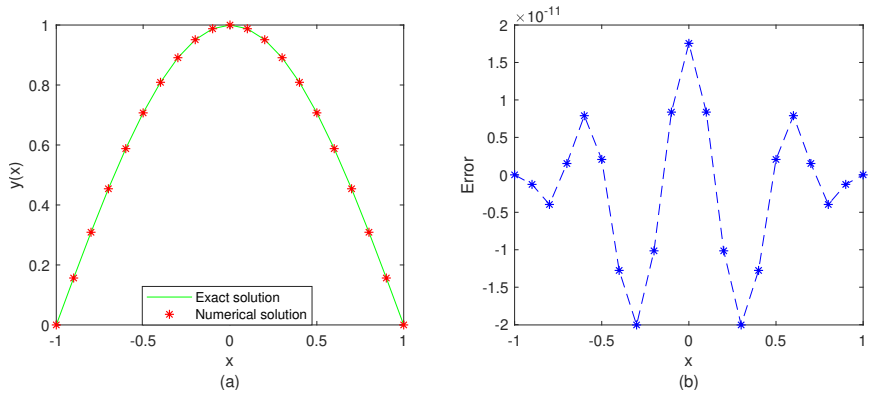

Fig. 2: The exact solution, the approximate solution and the errors for BVPs of fourth-order linear ODE for Example 1

x	Exact solution	HNN solution	Absolute error by HNN method	Absolute error by LS-SVM method [25]
−0.815	0.286524552727799	0.286524552727795	0.40e-11	7.25e-10
−0.630	0.549022817998132	0.549022817991267	0.69e-11	1.23e-09
−0.445	0.765483213493088	0.765483213499043	0.60e-11	2.39e-09
−0.260	0.917754625683981	0.917754625702042	1.81e-11	2.43e-09
−0.075	0.993068456954926	0.993068456942747	1.22e-11	1.48e-09
0.1100	0.985109326154774	0.985109326148116	0.67e-11	1.62e-09
0.2950	0.894544639838025	0.894544639857977	2.00e-11	2.56e-09
0.4800	0.728968627421412	0.728968627422055	0.06e-11	2.21e-09
0.6650	0.502265533143373	0.502265533138883	0.45e-11	1.06e-09
0.8500	0.233445363855905	0.233445363859109	0.32e-11	6.35e-10

Table 2: Comparison among the exact solution , HNN solution and LS-SVM solution (Example 1)

In Tab. 2, ten test points at unequal intervals in the domain $[-1,1]$ are adopted, clearly, the proposed method is compared with LS-SVM algorithm, which shows very good accuracy for solving BVPs of higher order linear ODEs. Note that the maximum absolute error is approximately 2.56×10^{-9} in [25], the maximum absolute error is approximately 2.00×10^{-11} by the proposed algorithm.

4.2 Example 2

Consider a eighth-order boundary value problem [41]

$$\begin{cases} \dfrac{d^8y}{dx^8} + xy(x) = -(48+15x+x^3)e^x, \quad x \in (0,1], \\ y(0) = 0,\ y(1) = 0, \\ y'(0) = 1,\ y'(1) = -e, \\ y^{(2)}(0) = 0,\ y^{(2)}(1) = -4e, \\ y^{(3)}(0) = -3,\ y^{(3)}(1) = -9e. \end{cases} \tag{23}$$

The analytical solution is $y = x(1-x)e^x$.

When 21 equidistant points are used for the training points and the first eight Hermite polynomials are used as basis functions, the approximate solution is described in Fig. 3 (a), and the error is shown in Fig. 3 (b). The MAE is approximately 5.97×10^{-15} and the MSE is approximately 4.44×10^{-30}. Furthermore, we respectively train the proposed HNN model using 21 and 101 equidistant points under 5 hidden neurons and 10 hidden neurons in the domain $[0,1]$. Table 9 shows the MSE and MAE of example 2 with different numbers of training points and hidden neurons.

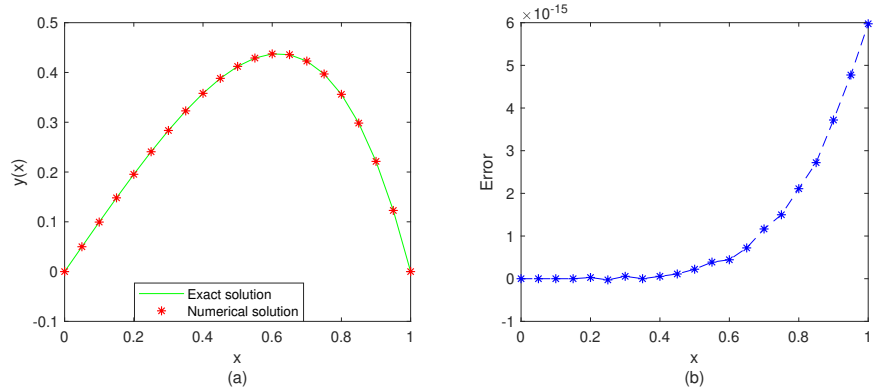

Fig. 3: The exact solution, the approximate solution and the errors for BVPs of eighth-order linear ODE for Example 2

We have compared HNN solution with non-polynomial spline solution in [41], and it is apparent that our proposed method outperforms the method in [41] in terms of accuracy from Tab. 3. Indeed, the accuracy of the error in [41] is $O(10^{-9})$, but the accuracy of the error obtained by our proposed method is $O(10^{-15})$ (Note that [41], the maximum absolute error is approximately 2.71×10^{-8}).

x	Exact solution	HNN solution	Absolute error by HNN method	Absolute error by non-polynomial spline [41]
0.1	0.099465382626808	0.099465382626808	0	5.62e-10
0.2	0.195424441305627	0.195424441305627	0.28e-16	4.88e-9
0.3	0.283470349590961	0.283470349590961	0.56e-16	1.37e-8
0.4	0.358037927433905	0.358037927433905	0.56e-16	2.29e-8
0.5	0.412180317675032	0.412180317675032	2.22e-16	2.71e-8
0.6	0.437308512093722	0.437308512093722	4.44e-16	2.38e-8
0.7	0.422888068568800	0.422888068568799	1.17e-15	1.49e-8
0.8	0.356086548558795	0.356086548558793	2.11e-15	5.54e-9

Table 3: Comparison among the exact solution , HNN solution and non-polynomial spline solution (Example 2)

4.3 Example 3

Consider a third-order linear ODE with initial conditions [42]

$$\begin{cases} \dfrac{d^3y}{dx^3} = 3sin(x), \quad x \in [0,1], \\ y(0) = 1, \\ y'(0) = 0, \\ y''(0) = -2. \end{cases} \tag{24}$$

The analytical solution is $y = 3cos(x) + x^2/2 - 2$.

The HNN algorithm for IVP of third-order linear ODE has been trained with 21 equidistant points in the domain $[0,1]$ and eight neurons. Comparison between the true solution and the approximate solution via our proposed HNN algorithm is given in Fig. 4 (a). Plot of the error function is cited in Fig. 4 (b), where the MSE and MAE are approximately 2.79×10^{-25} and 1.11×10^{-12}.

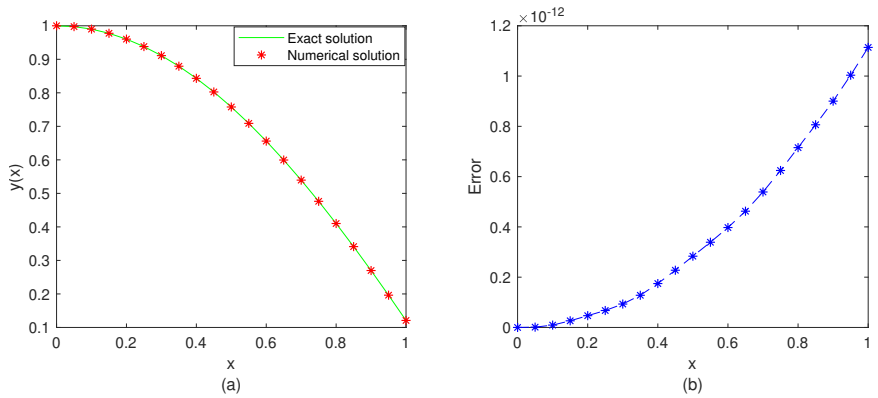

Fig. 4: The exact solution, the approximate solution and the errors for IVPs of third-order linear ODE for Example 3

Nonuniform partitions of $[0,1]$ using ten points are used for test process. Comparison among the exact solution, HNN solution and R-K solution is shown for example 3 in Tab. 4, from which we can see that the maximum absolute error by HNN method is approximately 1.09×10^{-12} and the maximum absolute error by R-K method is approximately 0.15×10^{-8}. It is clear that the approximate solution obtained by HNN method is of higher accuracy compared to the solution obtained by R-K method.

x	Exact solution	HNN solution	Absolute error by HNN method	Absolute error by R-K method
0.0000	1.000000000000000	1.000000000000000	0	0
0.0949	0.991004125483915	0.991004125483907	0.01e-12	0.01e-8
0.1546	0.976170191314165	0.976170191314136	0.03e-12	0.04e-8
0.2658	0.929972813978730	0.929972813978655	0.07e-12	0.02e-8
0.3179	0.900211947932621	0.900211947932517	0.10e-12	0.01e-8
0.4865	0.770265049717399	0.770265049717131	0.27e-12	0.05e-8
0.7289	0.503370128894918	0.503370128894330	0.59e-12	0.10e-8
0.8345	0.362839866009161	0.362839866008383	0.78e-12	0.15e-8
0.9035	0.264749691981959	0.264749691981051	0.91e-12	0.13e-8
0.9876	0.139761102333956	0.139761102332870	1.09e-12	0.04e-9

Table 4: Comparison among the exact solution, HNN solution and R-K solution (Example 3)

4.4 Example 4

Consider a third-order linear ordinary differential equation with initial conditions [42]

$$\begin{cases} \dfrac{d^3y}{dx^3} = -e^{-x}, \quad x\in(0,1], \\ y(0)=1, \\ y'(0)=-1, \\ y''(0)=1. \end{cases} \tag{25}$$

The analytical solution is $y=e^{-x}$.

The approximate solution obtained by HNN algorithm is compared with the exact solution and results are shown in Fig. 5 (a). The absolute errors for twenty-one equidistant points in the domain $[0,1]$ are depicted in Fig. 5 (b). Furthermore, the results of the approximate solution via our proposed HNN algorithm and R-K method for ten non-equidistant test points are shown in Table 5, in which we can see that the maximum absolute error obtained by HNN method is approximately 2.57×10^{-13} and the maximum absolute error obtained by R-K method is approximately 5.37×10^{-10}. It is obvious that the proposed method is better than R-K method in terms of accuracy.

Table 9 lists the MSE and MAE of example 4 with different numbers of training points and hidden neurons. It is apparent that our proposed method still has a better performance in accuracy even with fewer training points.

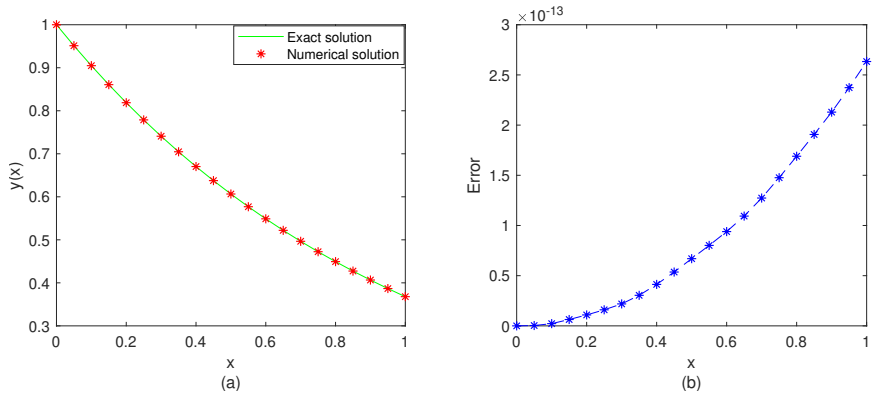

Fig. 5: The exact solution, the approximate solution and the errors for IVPs of third-order linear ODE for Example 4

x	Exact solution	HNN solution	Absolute error by HNN method	Absolute error by R-K method
0.0000	1.000000000000000	1.000000000000000	0	0
0.0949	0.909463876308695	0.909463876308693	0.02e-13	0.01e-10
0.1546	0.856757812076957	0.856757812076951	0.07e-13	3.34e-10
0.2658	0.766592430656324	0.766592430656306	0.18e-13	5.00e-10
0.3179	0.727675552331572	0.727675552331547	0.25e-13	5.37e-10
0.4865	0.614774343282285	0.614774343282222	0.63e-13	0.13e-10
0.7289	0.482439381641174	0.482439381641035	1.39e-13	2.40e-10
0.8345	0.434091472917058	0.434091472916874	1.84e-13	0.88e-10
0.9035	0.405149153267934	0.405149153267720	2.15e-13	3.07e-10
0.9876	0.372469546077995	0.372469546077739	2.57e-13	0.36e-10

Table 5: Comparison among the exact solution, HNN solution and R-K solution (Example 4)

4.5 Example 5

Consider a seventh-order linear ordinary differential equation [43]

$$\frac{d^7y}{dx^7} - y = -e^x(14x+35), \quad x \in [0,1], \tag{26}$$

subject to initial conditions $y(0)=0, y'(0)=1, y^{(2)}(0)=0, y^{(3)}(0)=-3, y^{(4)}(0)=-8, y^{(5)}(0)=-15, y^{(6)}(0)=-24$. The analytical solution is $y=x(1-x)e^x$.

The first eight Hermite polynomials are used as basis functions and twenty-one equidistant points are chosen for training HNN model. Comparison between analytical and HNN results is depicted in Fig. 6 (a), the error function between analytical and HNN results is cited in Fig. 6 (b), where the maximum absolute error is approximately 0.44×10^{-12} and the MSE is approximately 2.04×10^{-26}.

Finally, comparison among exact, HNN and R-K results for some testing points is shown in Table 6. The accuracy of the error obtained by R-K method is $O(10^{-6})$, while the accuracy of the error obtained by our proposed method is $O(10^{-15})$, the advantage of the HNN method can be seen particularly.

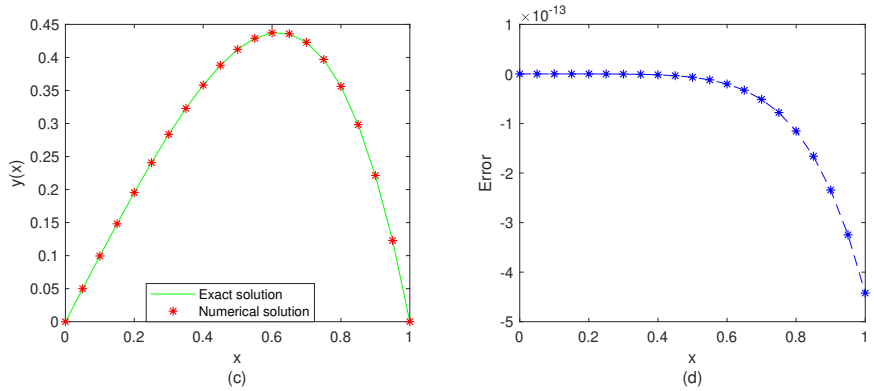

Fig. 6: The exact solution, the approximate solution and the errors for IVPs of seventh-order linear ODE for Example 5

x	Exact solution	HNN solution	Absolute error by HNN method	Absolute error by R-K method
0.0000	0	0	0	0
0.0949	0.094444641769197	0.094444641769197	0	2.00e-6
0.1546	0.152550508624087	0.152550508624087	0.03e-15	2.21e-5
0.2658	0.254568597596144	0.254568597596145	0.17e-15	3.32e-4
0.3179	0.297989384561865	0.297989384561865	0.39e-15	8.12e-4
0.4865	0.406356824629702	0.406356824629708	5.66e-15	6.81e-3
0.7289	0.409595065244847	0.409595065244912	0.66e-13	5.14e-2
0.8345	0.318158173142435	0.318158173142584	1.49e-13	1.01e-1
0.9035	0.215199141592037	0.215199141592277	2.40e-13	1.51e-1
0.9876	0.032878500078596	0.032878500079006	4.10e-13	2.35e-1

Table 6: Comparison among the exact solution, HNN solution and R-K solution (Example 5)

4.6 Example 6

Consider system of third-order linear ordinary differential equations [13]

$$\begin{cases} \dfrac{d^3y_1(x)}{dx^3} = \dfrac{1}{68}\big(817y_1(x)+1393y_2(x)+448y_3(x)\big), \quad x\in[0,1], \\ \dfrac{d^3y_2(x)}{dx^3} = -\dfrac{1}{68}\big(1141y_1(x)+2837y_2(x)+896y_3(x)\big), \\ \dfrac{d^3y_3(x)}{dx^3} = \dfrac{1}{136}\big(3059y_1(x)+4319y_2(x)+1592y_3(x)\big), \end{cases} \tag{27}$$

subject to initial conditions

$$\begin{cases} y_1(0)=2, y_2(0)=-2, y_3(0)=-12, \\ y_1'(0)=-12, y_2'(0)=28, y_3'(0)=-33, \\ y_1''(0)=20, y_2''(0)=-52, y_3''(0)=5. \end{cases} \tag{28}$$

The analytical solutions are given by

$$\begin{cases} y_1(x) = e^x - 2e^{2x} + 3e^{-3x}, \\ y_2(x) = 3e^x + 2e^{2x} - 7e^{-3x}, \\ y_3(x) = -11e^x - 5e^{2x} + 4e^{-3x}. \end{cases} \tag{29}$$

The HNN models for IVP of system of third-order linear ordinary differential equations have been trained with 21 equidistant points in the domain $[0,1]$ and the first eight Hermite polynomials. From Fig. 7, we can easily observe that the results from the HNN models coincide with those from the exact results, moreover, the mean square error of $y_1(x)$ is 3.04×10^{-15}, the mean square error of $y_2(x)$ is only 1.79×10^{-14}, and the mean square error of $y_3(x)$ is 4.63×10^{-15} .

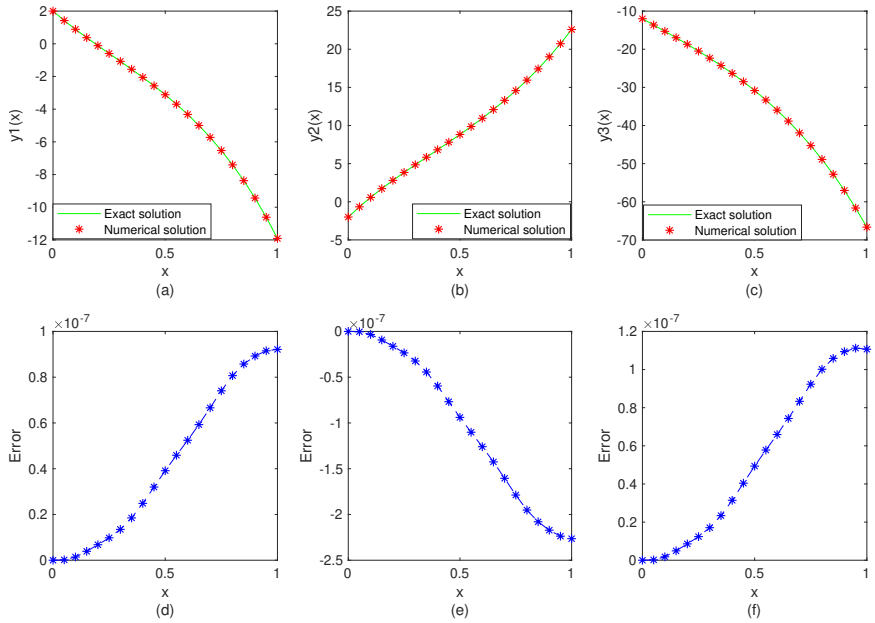

Fig. 7: The exact solutions, the approximate solutions and the errors for IVPs of system of third–order linear ordinary differential equations for Example 6

Table 7 lists the absolute errors of the approximate solutions obtained by our proposed HNN models for some random test points, the accuracy of the errors obtained by our proposed method is $O(10^{-7})$, which shows that our method has good generalization ability. Moreover, we list the MSE and MAE of the numerical solutions obtained by HNN method when more training points are added and different numbers of hidden layer neurons are configured in Table 9.

x	Absolute error$(y_1(x))$	Absolute error $(y_2(x))$	Absolute error $(y_3(x))$
0.0915	0.11e-08	0.03e-07	0.01e-07
0.1518	0.40e-08	0.10e-07	0.05e-07
0.2410	0.92e-08	0.22e-07	0.12e-07
0.3604	1.97e-08	0.47e-07	0.25e-07
0.5267	4.28e-08	1.03e-07	0.54e-07
0.6395	5.78e-08	1.39e-07	0.73e-07
0.7590	7.53e-08	1.82e-07	0.94e-07
0.8482	8.56e-08	2.08e-07	1.06e-07
0.9134	9.00e-08	2.19e-07	1.10e-07
0.9784	9.22e-08	2.26e-07	1.11e-07

Table 7: Comparison between exact solutions and HNN solutions (Example 6)

4.7 Example 7

Consider system of second-order linear ordinary differential equations [44]

$$\begin{cases} \dfrac{d^2y_1(x)}{dx^2} + \dfrac{dy_1(x)}{dx} + xy_1(x) + \dfrac{dy_2(x)}{dx} + 2xy_2(x) = f_1(x), \\ \dfrac{d^2y_2(x)}{dx^2} + y_2(x) + 2\dfrac{dy_2(x)}{dx} + x^2y_1(x) = f_2(x) \end{cases} \tag{30}$$

where $f_1(x) = -2(1+x)cos(x) + \pi cos(\pi x) + 2xsin(\pi x) + (4x - 2x^2 - 4)sin(x)$, and $f_2(x) = 4(1-x)cos(x) + 2(x^2 - x^3 - 2)sin(x) + (1-\pi^2)sin(\pi x)$,
subject to boundary conditions

$$\begin{cases} y_1(0) = y_2(0) = 0, \\ y_1(1) = y_2(1) = 0, \end{cases} \tag{31}$$

The analytical solutions are given by

$$\begin{cases} y_1(x) = 2(1-x)sin(x), \\ y_2(x) = sin(\pi x), \end{cases} \tag{32}$$

The exact solutions and the obtained solutions via our proposed HNN models are plotted in Fig. 8 (a) and (c), the error plots between analytical and HNN solutions are cited in Fig. 8 (b) and (d), where twenty-one equidistant points in the interval $[0,1]$ are utilized for the training process.

Moreover, Table 8 lists results of the exact solutions and the approximate solutions by our proposed HNN models for ten random testing points, the accuracy of the error of $y_1(x)$ is $O(10^{-11})$ and the accuracy of the error of $y_2(x)$ is $O(10^{-10})$. All of this said that the better generalization ability of HNN even under the conditions of limited training samples.

Table 9 compares the errors of the numerical solutions obtained by HNN models when more training points are added and different numbers of hidden layer neurons are configured. From these results, it can be seen that the largest absolute errors are not beyond 10^{-4} magnitude order, which show that the proposed HNN algorithm could reach a quite agreeable accuracy. Therefore, we can conclude that the proposed

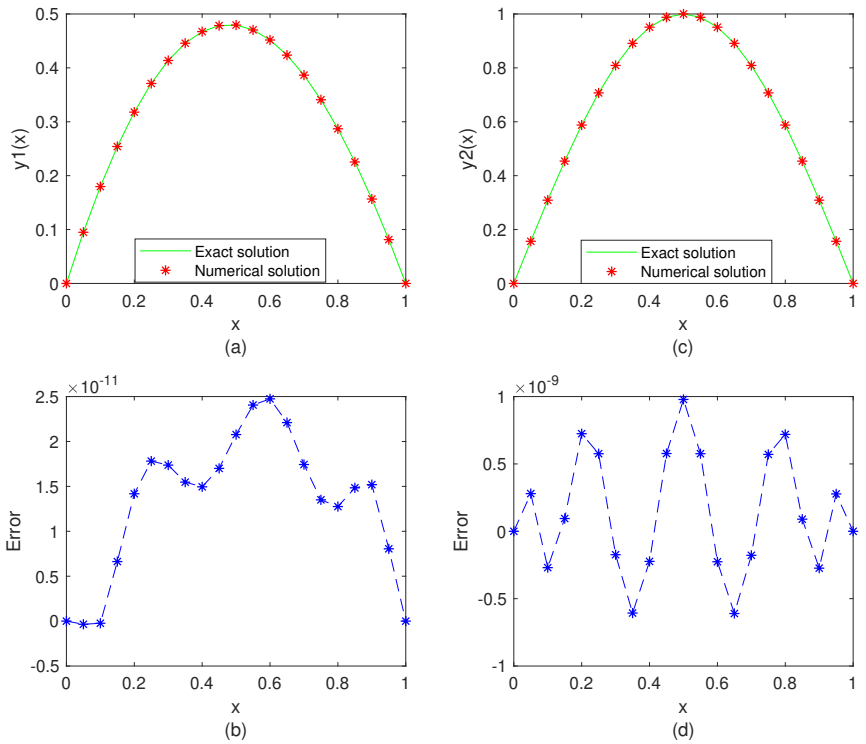

Fig. 8: The exact solutions, the approximate solutions and the errors for BVPs of system of second–order linear ordinary differential equation for Example 7

x	Absolute error$(y_1(x))$	Absolute error $(y_2(x))$
0.0915	0.08e-11	2.25e-10
0.1518	0.69e-11	1.21e-10
0.2410	1.75e-11	6.69e-10
0.3604	1.52e-11	5.94e-10
0.5267	2.27e-11	8.56e-10
0.6395	2.29e-11	5.97e-10
0.7590	1.31e-11	6.64e-10
0.8482	1.48e-11	1.16e-10
0.9134	1.41e-11	1.88e-10
0.9784	0.19e-11	3.51e-10

Table 8: Comparison between exact solutions and HNN solutions (Example 7)

HNN algorithm is feasible in solving numerically IVPs and BVPs of higher order linear ODEs and higher order linear SODEs.

Example	MSE			MAE		
	L=20,N=5	L=20,N=10	L=100,N=10	L=20,N=5	L=20,N=10	L=100,N=10
Example1	1.98e-12	2.85e-28	4.96e-28	2.87e-06	4.29e-14	4.79e-14
Example2	1.04e-24	4.44e-30	3.61e-30	1.95e-12	5.97e-15	5.97e-15
Example3	7.42e-17	1.95e-31	4.83e-32	1.81e-08	9.99e-16	6.66e-16
Example4	1.67e-17	1.57e-32	5.24e-33	8.59e-09	2.22e-16	2.22e-16
Example5	9.32e-18	2.63e-32	3.25e-32	9.48e-09	4.39e-16	1.66e-16
$\mathbf{Ex.6}(\mathbf{y_1}(\mathbf{x}))$	9.04e-09	1.53e-19	5.86e-21	1.72e-04	0.67e-09	0.13e-09
$\mathbf{y_2}(\mathbf{x})$	3.70e-08	8.61e-19	3.30e-20	3.38e-04	1.59e-09	0.32e-09
$\mathbf{y_3}(\mathbf{x})$	2.21e-08	2.55e-19	9.75e-21	2.78e-04	0.85e-09	0.17e-09
$\mathbf{Ex.7}(\mathbf{y_1}(\mathbf{x}))$	8.87e-13	3.19e-26	7.15e-30	1.71e-06	2.52e-13	4.75e-15
$\mathbf{y_2}(\mathbf{x})$	3.61e-10	4.62e-24	2.25e-24	3.31e-05	3.59e-12	2.77e-12

Table 9: Comparison of the errors of different example with different numbers of training points and hidden neurons.

5 Conclusion

In this paper, the Hermite neural network algorithm with improved extreme learning machine has been developed to solve IVPs and BVPs of higher order linear ODEs and higher order linear SODEs. Our proposed HNN model can directly solve the initial and boundary value problems of higher order linear ODEs and higher order linear SODEs with high accuracy. At the same time, this paper compares HNN model with least squares support vector machines method (LS-SVM method), non-polynomial spline method and Runge-Kutta method (R-K method).

Experiments on the common and random data sets indicate that the HNN model achieves much higher accuracy, lower complexity but stronger generalization ability than existed methods. Evidently, the proposed HNN algorithm could be a good tool to solve higher order linear ODEs and higher order linear SODEs.

Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities of Central South University (Grant No.2021zzts0046).

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. W. D. S. Clarence, Modeling and Control of Engineering Systems, CRC Press, 2009.
2. B. Kazmierczak and T. Lipniacki, Homoclinic solutions in mechanical systems with small dissipation, Application to DNA dynamics, Journal of Mathematical Biology, **44** (2002), 309–329.
3. Z. A. Majid, N. A. Azmi, M. Suleiman and Z. B. Ibrahaim, Solving directly general third order ordinary differential equations using two-point four step block method, Sains Malaysiana, **41** (2012), 623–632.

4. K. Hussain, F. Ismail, N. Senu and F. Rabiei, Fourth–order improved Runge-Kutta method for directly solving special third-order ordinary differential equations , Iranian Journal of Science and Technology Transaction A-Science, **41** (2017), 429–437.
5. K. Hussain, F. Ismail and N. Senu, Solving directly special fourth-order ordinary differential equations using Runge-Kutta type method , Journal of Computational and Applied Mathematics, **306** (2016), 179–199.
6. S. Islam, I. Aziz and B. Šarler, The numerical solution of second-order boundary value problems by collocation method with the Haar wavelets, Mathematical and Computer Modelling, **52** (2010), 1577–1590.
7. R. Amin, K.Shah, Q. M. Al-Mdallal, I. Khan and M. Asif, Efficient Numerical Algorithm for the Solution of Eight Order Boundary Value Problems by Haar Wavelet Method, International Journal of Applied and Computational Mathematics, (2021) 7–34.
8. Z. Shi and F. Li, Numerical solution of high-order differential equations by using periodized Shannon wavelets, Applied Mathematical Modelling, **38** (2014), 2235–2248.
9. W. Al-Hayani, Adomian decomposition method with Green's function for sixth-order boundary value problems, Computers and Mathematics with Applications, **61** (2011), 1567–1575.
10. E. H. Doha and W. M. Abd-Elhameed, Efficient spectral-Galerkin algorithms for direct solution of second-order equations using ultraspherical polynomials, SIAM Journal on Scientific Computing, **24** (2002), 548–571.
11. E. H. Doha and W. M. Abd-Elhameed, Efficient spectral ultraspherical-dual-Petrov-Galerkin algorithms for the direct solution of (2n+1)th-order linear differential equations, Mathematics and Computers in Simulation, **79** (2009), 3221–3242.
12. E. H. Doha, W. M. Abd-Elhameed, and Y. H. Youssri, Efficient spectral-Petrov-Galerkin methods for the integrated forms of third- and fifth-order elliptic differential equations using general parameters generalized Jacobi polynomials, Applied Mathematics and Computation, **218** (2012), 7727–7740.
13. X. You and Z. Chen, Direct integrators of Runge-Kutta type for special third-order ordinary differential equations, Applied Numerical Mathematics, **74** (2013), 128–150.
14. M. Dehghan and A. Nikpour, Numerical solution of the system of second-order boundary value problems using the local radial basis functions based differential quadrature collocation method, Applied Mathematical Modelling, **37** (2013), 8578–8599.
15. N. Caglar and H. Caglar, B-spline method for solving linear system of second-order boundary value problems, Computers and Mathematics with Applications, **57** (2009), 757–762.
16. M. B. Abdulla, A. L. Costa and R. L. Sousa, Probabilistic identification of subsurface gypsum geohazards using artificial networks, Neural Computing and Applications, **29** (2018), 1377–1391.
17. F. Weng, Y. Chen, Z. Wang, M. Hou, J. Luo and Z. Tian, Gold price forecasting research based on an improved online extreme learning machine algorithm, Journal of Ambient Intelligence and Humanized Computing, **11** (2020), 4101–4111.
18. N. Mai-Duy, Solving high order ordinary differential equations with radial basis function networks, International Journal for Numerical Methods in Engineering, **62** (2005), 824–852.
19. A. Malek and R. S.Beidokhti, Numerical solution for high order differential equations using a hybrid neural network-optimization method, Applied Mathematics and Computation, **183** (2006), 260–271.
20. H. S. Yazdi and R. Pourreza, Unsupervised adaptive neural-fuzzy inference system for solving differential equations, Applied Soft Computing, **10** (2010), 267–275.
21. H. S. Yazdi, M. Pakdaman and H. Modaghegh, Unsupervised kernel least mean square algorithm for solving ordinary differential equations, Neurocomputing, **74** (2011), 2062–2071.
22. H. S. Yazdi, H. Modaghegh and M. Pakdaman, Ordinary differential equations solution in kernel space, Neural Computing and Applications, **21** (2012), S79–S85.
23. S. Mall and S. Chakraverty, Application of Legendre neural network for solving ordinary differential equations, Applied Soft Computing, **43** (2016), 347–356.
24. S. Mall and S. Chakraverty, Chebyshev neural network based model for solving Lane-Emden type equations, Applied Mathematics and Computation, **247** (2014), 100–114.
25. Y. Lu, Q. Yin, H. Li, H. Sun, Y. Yang and M. Hou. The LS-SVM algorithms for boundary value problems of high-order ordinary differential equations, Advances in Difference Equations, **195** (2019),1–22.
26. G. B. Huang, Q. Y. Zhu and C. K. Siew, Extreme learning machine: theory and applications, Neurocomputing, **70** (2006), 489-501.
27. G. B. Huang, H. Zhou, X. Ding and R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, **42** (2012), 513-529.

28. E. Cambria and G. B. Huang, Extreme learning machine: trends and controversies, IEEE Intelligent Systems, **28** (2013), 30–59.
29. C. Wong, C. Vong, P. Wong and J. Cao, Kernel-based multilayer extreme learning machines for representation learning, IEEE Transactions on Neural Networks and Learning Systems, **29** (2018), 757–762.
30. K. Rudd and S. Ferrari, A constrained integration (CINT) approach to solving partial differential equations using artificial neural networks, Neurocomputing, **155** (2015), 277–285.
31. C. J. Zuniga–Aguilar, H. M. Romero-Ugalde, J. F. Gomez–Aguilar, R. F. Escobar-Jimenez and M. Valtierra-Rodrłguez, Solving fractional differential equations of variable-order involving operators with Mittag-Leffler kernel using artificial neural networks, Chaos Solitons and Fractals, **103** (2017), 382–403.
32. F. Rostami and A. Jafarian, A new artificial neural network structure for solving high-order linear fractional differential equations, International Journal of Computer Mathematics, **95** (2018), 528–539.
33. M. Pakdaman, A. Ahmadian, S. Effati, S. Salahshour and D. Baleanu, Solving differential equations of fractional order using an optimization technique based on training artificial neural network, Applied Mathematics and Computation, **293** (2017), 81–95.
34. Y. Yang, M. Hou and J. Luo, A novel improved extreme learning machine algorithm in solving ordinary differential equations by Legendre neural network methods, Advances in Difference Equations, **469** (2018), 1–24.
35. Y. Yang, M. Hou, H. Sun, T. Zhang, F. Weng and J. Luo, Neural network algorithm based on Legendre improved extreme learning machine for solving elliptic partial differential equations, Soft Computing, **24** (2020), 1083–1096.
36. H. Sun, M. Hou, Y. Yang, T. Zhang, F. Weng and F. Han, Solving partial differential equation based on Bernstein neural network and extreme learning machine algorithm, Neural Processing Letters, **50** (2019), 1153–1172.
37. Y. Cheng, M. Hou and J. Wang, An improved optimal trigonometric ELM algorithm for numerical solution to ruin probability of Erlang(2) risk model Multimedia Tools and Applications, **79** (2020), 30235–30255.
38. Y. Lu, G. Chen, Q. Yin, H. Sun and M. Hou, Solving the ruin probabilities of some risk models with Legendre neural network algorithm, Digital Signal Processing, **99** (2020), 102634.
39. Y. Chen, H. Yu, X. Meng , X. Xie , M. Hou and J. Chevallier, Numerical solving of the generalized Black-Scholes differential equation using Laguerre neural network, Digital Signal Processing, **112** (2021), 103003.
40. G. E. Andrews, R. Askey and R. Roy, Special functions, Tsinghua University Press, 2004.
41. S. S. Siddiqi and G. Akram, Solution of eighth-order boundary value problems using the nonpolynomial spline technique, International Journal of Computer Mathematics, **84** (2007), 347–368.
42. Z. A. Majid, M. Suleiman and N. A. Azmi, Variable step size block method for solving directly third order ordinary differential equations, Far East Journal of Mathematical Sciences, **41** (2010), 63–73.
43. Y. H. Youssri, W. M. Abd-Elhameed and M. Abdelhakem, A robust spectral treatment of a class of initial value problems using modified Chebyshev polynomials, Mathematical Methods in the Applied Sciences, **44** (2021), 9224–9236.
44. F. Geng and M. Cui, Solving a nonlinear system of second order boundary value problems, Journal of Mathematical Analysis and Applications, **327** (2007), 1167–1181.